12.12.12

Constructors

Frederik Drosdzol

dkfz. GERMAN CANCER RESEARCH CENTER IN THE HELMHOLTZ ASSOCIATION

Constructors

What the compiler automatically adds to your class

- Default-constructor
- Copy-constructor
- Assignment-operator

Default-Constructor

- On object creation, compiler automatically calls constructor
- If no constructor is defined, compiler will generate a parameterless “default-constructor”
- Notice, user generated parameterless constructor is also called „default-constructor

```
class IPhone {
  std::string m_name;
};

int main()
{
  IPhone iPhone5;    // Object created and default-constructor called automatically
                     // Initializing iPhone5.m_name with an empty string
}
```

Default-Constructor

- On object creation, compiler automatically calls constructor
- If no constructor is defined, compiler will generate a parameterless “default-constructor”
- Notice, user generated parameterless constructor is also called „default-constructor

```
class IPhone {
  std::string m_name;
public:
  IPhone()    // Inline compiler generated default-constructor
  : m_name()    // Default-constructor from member
  {}
};

int main()
{
  IPhone iPhone5;   // Object created and default-constructor called automatically
                    // Initializing iPhone5.m_name with an empty string
}
```

Inhibition of default constructor

- Consider: If user declares constructor with parameters, compiler delivers no constructor

```
class IPhone {
public:
  IPhone(std::string name)
    : m_name(name)
  {}
private:
  std::string m_name;
};

int main() {
  IPhone iPhone5("an iPhone");
  IPhone iPhone4S;    // no default-constructor created
}
```

Error:

```
mbimac01:Desktop drosdzol$ g++ constructor.cpp -o constructor
constructor.cpp: In function 'int main()':
constructor.cpp:18: error: no matching function for call to 'IPhone::IPhone()'
constructor.cpp:8: note: candidates are: IPhone::IPhone(std::string)
constructor.cpp:6: note:                 IPhone::IPhone(const IPhone&)
```

Inhibition of default constructor

- Subclass calls constructor of upper class
- Call of parameterless Smartphone() not working

```
class Smartphone {
public:
  Smartphone(std::string name)
  : m_name(name);
  {}
private:
  std::string m_name;
};

class IPhone: public Smartphone {
public:
  IPhone(std::string name)
  : Smartphone("some text")          // Smartphone() not working
  {}
private:
};

int main() {
  IPhone iPhone5("myiPhone");
}
```

Initialization-list

- Initialization-list is faster (because no default-constructor is invoked for objects)
- So instead of:

```
IPhone::IPhone()
{
  m_name = "an iPhone";
}
```

- Use:

```
IPhone::IPhone()
: m_name("an iPhone")
{
}
```

- Different order of declaration -> warning

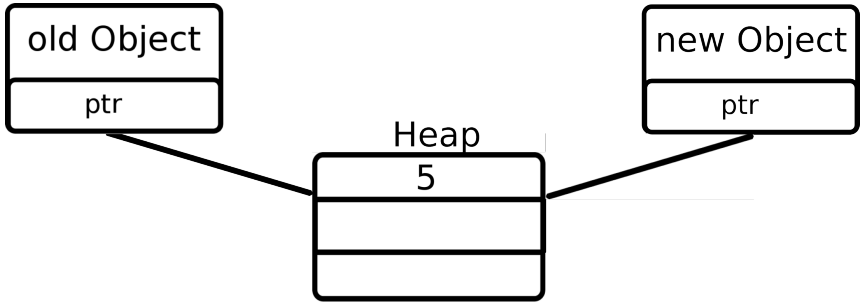
Copy-Constructor

- Compiler will generate copy-constructor
- Copy-constructor always called when a copy of an object is created (e.g. parameter or return-value)
- Copy-constructor and assignment-operator copy data contained in other objects to the data members of the current objects

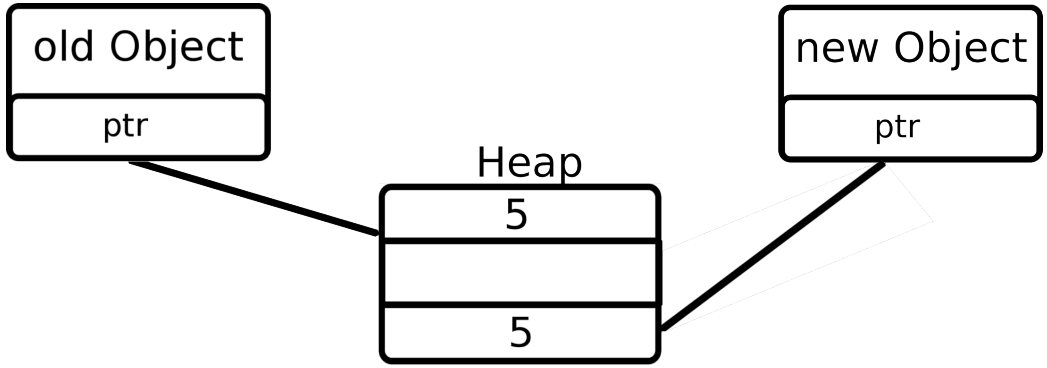
```
class IPhone {
public:
  IPhone(const IPhone & rv):
    value(rv.value)
  {}
private:
  int value;
};

int main(){
  IPhone iPhone5("new iPhone");
  IPhone iPhone4S(iPhone5);   // copy-constructor
}
```

Copy-Construcor

- Elementwise copy of member variables
- Problem with pointer: When destructor is called, object is removed from memory but other pointer still pointing to memory



- Solution: Copy-constructor which allocates memory and copies values to new memory (deep copy)



Copy-Construcor

- Example:

```
class IPhone {
public:
  IPhone();
  IPhone(const IPhone &);        // copy-constructor
  void setValue(int tmp) {*value = tmp;}
private:
  int *value;
}

IPhone::IPhone () {
  value = new int;
  *value= 0;
}

IPhone::IPhone(const IPhone & rhs) {
  value = new int;
  *value = *(rhs.value);
}

Int main() {
  IPhone iPhone4S;
  iPhone4S.setValue(42);
  IPhone iPhone5(iPhone4S);
}
```

Assignment-Operator

- Compiler will generate assignment-operator if no one is defined
- Assignment-operator is called when an object is assigned

```
IPhone iPhone5(42);
IPhone iPhone4S(21);
// some code
iPhone5 = iPhone4S;
```

- What if user makes `iPhone5 = iPhone5;`
- Assignment-operator checks this pointer

```
IPhone & IPhone::operator=(const IPhone & rhs) {
  if(this == &rhs)
    return *this;
  *value = *(rhs.value);
  return *this;
}
```

Outlook

What the compiler creates in the background

- Template Instantiations
- RTTI – Type_info